Resolução de problemas de programação com o método de composição de resultado

Rafael Gomes Sousa, Eloi Luiz Favero

Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Centro de Ciências Exatas e Naturais (CCEN) – Universidade Federal do Pará (UFPA)

{rafaelgomes, favero}@ufpa.br

Resumo. *O processo de ensino e aprendizagem de programação é uma tarefa bastante complexa, para tanto, defendemos a abordagem prática de resolução de problemas, para o melhor aproveitamento acadêmico dos estudantes. Entretanto, a construção da resposta para um problema possui uma alta carga de subjetividade. Aqui, concebemos um método resolução de problemas de programação (Composição de Resultado), auxiliado por um ferramental de apoio (Ambiente LabProg). E obtivemos melhorias expressivas na atuação de estudantes de programação em atividades práticas de laboratório, assim como, uma sensível variação nos conhecimentos teóricos.*

Palavras-chave*: treino de programação, ensino de programação, correção automatizada, método de resolução de problema.*

Solving programming problems with the outcome composition method

Abstract. *The teaching and learning of programming is a complex task, therefore, uphold the practical approach to problem solving, to the best academic achievement of students. However, the construction of the answer to a problem has a high load of subjectivity. Here, we designed a resolution method of scheduling problems (Result composition), aided by a tooling support (LabProg Environment). And we obtained significant improvements in performance programming students in laboratory practice activities as well as a substantial change in theoretical knowledge.*

Keywords*: programming training, educational programming, automated remediation, problem-solving method.*

1. O Ensino de Programação

No contexto do ensino de programação em cursos de graduação em computação, são apresentadas diversas propostas, com o objetivo de melhoria no processo de aquisição de conhecimento e habilidades práticas dos estudantes, dentre elas destacam-se a realização de exercícios práticos de resolução de problemas.

Dada a abordagem apresentada em Nobre e Menezes (2002), observa-se que na resolução de um determinado número de exercícios de programação, alguns alunos

encontram uma maneira de construir soluções, entretanto outros continuam estagnados, até mesmo, sem saber por onde começar. A situação dos alunos com dificuldade pode ser amenizada, por meio de um auxilio diferenciado dos professores dessas disciplinas introdutórias. Contudo, sabe-se que geralmente as turmas são numerosas e com diferentes perfis, dificultando a devida assistência individualizada.

Inseridos nessa problemática, Mota e Favero (2008), Mota et al. (2009) e Pelz (2012) propõem uma abordagem de correção automática das respostas de atividades de programação. Entretanto a avaliação é afetada pela alta complexidade da tarefa e a subjetividade empregada pelo professor.

Portanto, dada esta problemática, temos como foco a concepção de um método para resolver problemas de programação, aliado a um ambiente que venha a oferecer correção automatizada e oferta de feedbacks de auxílio aos estudantes. E deste modo, proporcionar um mecanismo sistematizado de resolução de atividade práticas. Para obter melhoria no desempenho dos estudantes, e facilitar o gerenciamento da interação entre professor e aluno.

A organização do trabalho segue pela devida revisão de trabalhos correlatos no item 2, amparando com ideias sobre tecnologias e melhores práticas, que venham a agregar valor. No item 3 é apresentado o método de resolução de problemas de programação. Na parte 4 o ambiente de apoio LabProg é exposto. O item 5 revela o delineamento experimental usado na realização dessa pesquisa. Na seção 6 são apresentados os principais resultados. Encerrando com as reflexões no item 7 considerações finais.

2. Trabalhos Relacionados

Em Mota et al. (2009) é relatado que o desenvolvimento de habilidades em programação exige um esforço significativo dos estudantes para construir soluções de problemas de programação. Ela apresenta um ambiente de exercício, simulação e avaliação para apoiar o processo ensino-aprendizagem em cursos de algoritmos e programação. Tudo isso integrado ao ambiente da plataforma MOODLE para potencializar os benefícios de sua utilização.

Em Rocha (2010) é apresentada a concepção e implementação de um método de ensino, baseado em um sistema personalizado que se adapta ao estudante, oferta conteúdo e avalia em níveis, assim o aluno avança quando não apresenta dificuldades. E a solução tecnológica foi integrado na plataforma MOODLE.

O trabalho de Pelz (2012) propõe um mecanismo para a correção automática de pequenos exercícios práticos de programação, através da verificação sintática, a verificação da presença de comandos obrigatórios, a verificação da adequação da estrutura do programa e a execução do programa para testar suas saídas. Assim, o mecanismo de correção automática foi utilizado em duas pesquisas distintas e apresentou bons resultados na geração de feedback para os aprendizes de programação.

No trabalho de Queirós e Leal (2012) é apresentada uma ferramenta para a assistência ao ensino de programação denominada PETCHA, que funciona de forma integrada a vários sistemas heterogêneos, aliando a correção automatizada e gerenciamento do repositório de problemas. Entretanto, apresenta como desvantagem a exibição dos testes, além do seu experimento não revelar ganhos estatísticos expressivos, contudo destaca benefícios na sua abordagem.

3. O Método de Ensino – Composição de Resultado

O objetivo da composição de resultado é oferecer um método de resolução de problemas de programação. Visto que o processo de resolução de um problema e bastante complexo e abstrato, e muitas vezes é definido como "criativo" Bennedsen e Caspersen (2004). Para tanto, este método oferece um conjunto de passos visando entender, iniciar e progredir até a resolução completa da resposta.

A Composição de Resultado foi concebida a partir de ideias como o *Test Driven Development* (TDD), o método de Divisão e Conquista e observações de como alunos resolviam problemas. O *Test Driven Development* é uma técnica de desenvolvimento de software baseada em um ciclo curto de repetições, em que são definidos teses, funcionalidades são criadas, assim como a realização de mudanças para aperfeiçoar o código, ambas avaliadas pelos testes Beck (2003).

Divisão e conquista é uma técnica para projeto de algoritmo, nela um problema é dividido em subproblemas menores do problema original. A conquista trata da resolução dos subproblemas se possível, caso contrário podem ser necessárias novas divisões. E por fim é realizada a combinação das soluções dos subproblemas para o problema original Cormen et al. (2009).

O método de Composição de resultado pode ser descrito nos seguintes passos:

1. Dividir o problema
2. Conceber a base do programa
3. Resolução dos subproblemas
   3.1. Construir a resposta para um subproblema
   3.2. Realizar um ou mais testes focados nesse subproblema
4. Integrar as respostas dos subproblemas

3.1. Passo 1: Dividir o problema

As abordagens tradicionais de ensino de programação não ajudam o estudante a construir a estrutura necessária para a resolução dos problemas. E muitos estudantes, nem sabem por onde começar, devido a enorme abstração entre a definição do problema e a implementação da resposta Bennedsen e Caspersen (2004).

A divisão do problema é orientada pelos possíveis desdobramentos que o problema engloba. Visto que, geralmente, os problemas de programação recebem valores como entrada, processam os dados e informam os resultados; conhecida como saída do programa.

Essa divisão ocorre analisando o resultado esperado, e o decompondo em subproblemas, ou seja, cada parte da saída pode se tornar um componente do problema original. Deste modo, realizando continuamente a resolução e combinação dos subproblemas, assim, a saída do problema vai ser composta e o problema integralmente resolvido.

3.2. Passo 2: Conceber a base do programa

A concepção da base do programa tem como objetivo estabelecer o mínimo de código necessário para dar início à resolução do problema. Portanto, este deve possuir as

variáveis mais básicas encontradas na descrição do problema, assim como apresentar como resultado valores triviais como resposta para o problema analisado.

Nessa etapa, o programa recebe uma entrada vazia, composta por valor (es) zero, devendo produzir uma resposta nula, com um ou mais valores zero, de acordo com a definição do problema. De tal forma, é esperado do estudante que ele possa produzir os códigos iniciais para um teste trivial. Além de poder testar entradas que de valores conhecidos, pois essa análise é importante para resolver partes do problema de forma isolada, pensando em entradas que favoreçam a resolução do subproblema.

3.3. Passo 3: Resolução dos subproblemas

Se em um dado problema possui uma entrada X e produz como resposta A, B e C, então neste ponto teremos a resposta "0 0 0" correspondente à saída esperada para um valor X=0. A resolução do subproblema que produz a saída A é menos complexa, pois nesse ponto estamos resolvendo apenas parte do problema.

Após tentar encontrar uma implementação para o subproblema A, um bom caso de teste deve ser usado para averiguar se esse subproblema foi resolvido. O candidato ideal para o teste seria um valor de X que consiga isolar A, ou seja, produzir resposta somente para A. Isto se faz necessário para segmentar completamente a resolução do problema e deixar o restante, B e C para os próximos passos.

3.4. Passo 4: Integrar as respostas dos subproblemas

A integração dos subproblemas já ocorre nas sucessivas execuções do Passo 3, visto que a cada nova parte do problema resolvido, irá progressivamente gerar a saída completa, assim como a resolução integrada do problema original. Entretanto, nesse passo é necessário averiguar se as partes integradas continuam apresentando respostas corretas para quaisquer valores de entrada.

3.5. Considerações sobre o método

A composição de resultado tenta segmentar a complexidade de problemas, o roteiro descrito pode ser expandido e adaptado, pois os problemas possuem diversos formatos e raciocínios associados. Portanto, problemas com várias entradas e saídas diferentes das apresentadas neste modelo podem se beneficiar deste método. Assim, focando nas ideias mais gerais que seriam: os desdobramentos do problema, a codificação básica e a construção e teste sucessivo.

4. Descrição do Ambiente

Conforme os problemas relacionados ao gerenciamento e retrabalho dos professores e alunos em uma disciplina de programação, foi realizado um trabalho de infraestrutura. Assim como agregando abordagens presentes na literatura, visando facilitar os trabalhos de correção das atividades e a oferta de feedbacks de ajuda, que proporcionem o auxílio do professor aos estudantes.

Foi construído um sistema web denominado LabProg – Laboratório de Programação, para o professor gerenciar os estudantes e suas atividades, listas e problemas. Já os estudantes visualizam as atividades pendentes de resolução e acompanham o seu desempenho registrado pelo sistema.

As principais funcionalidades podem ser ilustradas no diagrama da Figura 1, onde é destacado o que professores e alunos podem realizar na ferramenta de gerenciamento Web.

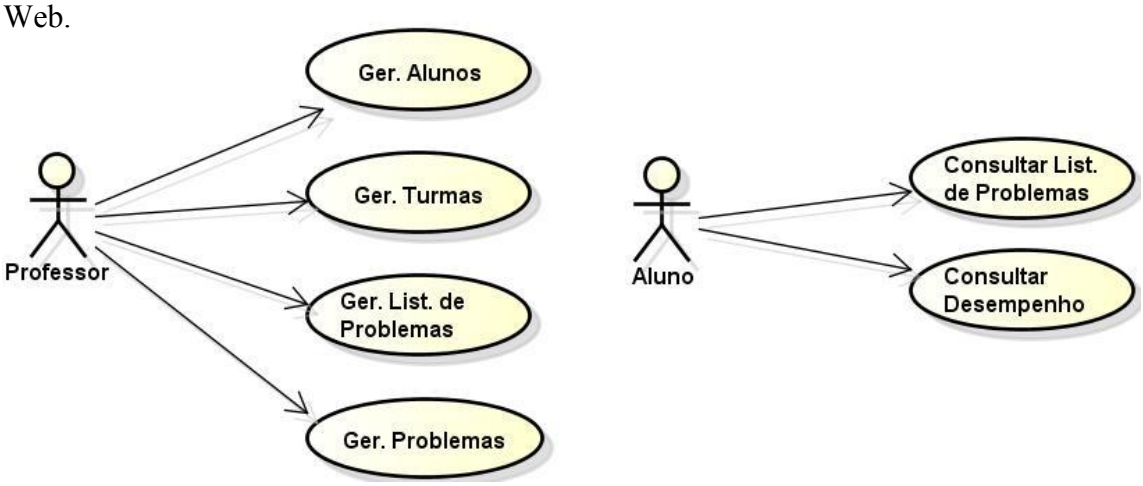

Figura 1. Principais funcionalidades do ambiente LabProg (Ger.=> Gerenciar).

Uma visão geral da interface do ambiente LabProg pode ser visualizado na Figura 2. Ilustrando a disposição dos menus e lista de problemas, mostrando as principais atividades de gerenciamento. As opções localizadas na lateral esquerda são separadas em grupos baseadas no tipo de usuário identificado pelo sistema. Neste contexto o exemplo da figura está exemplificando as opções de professor e aluno.

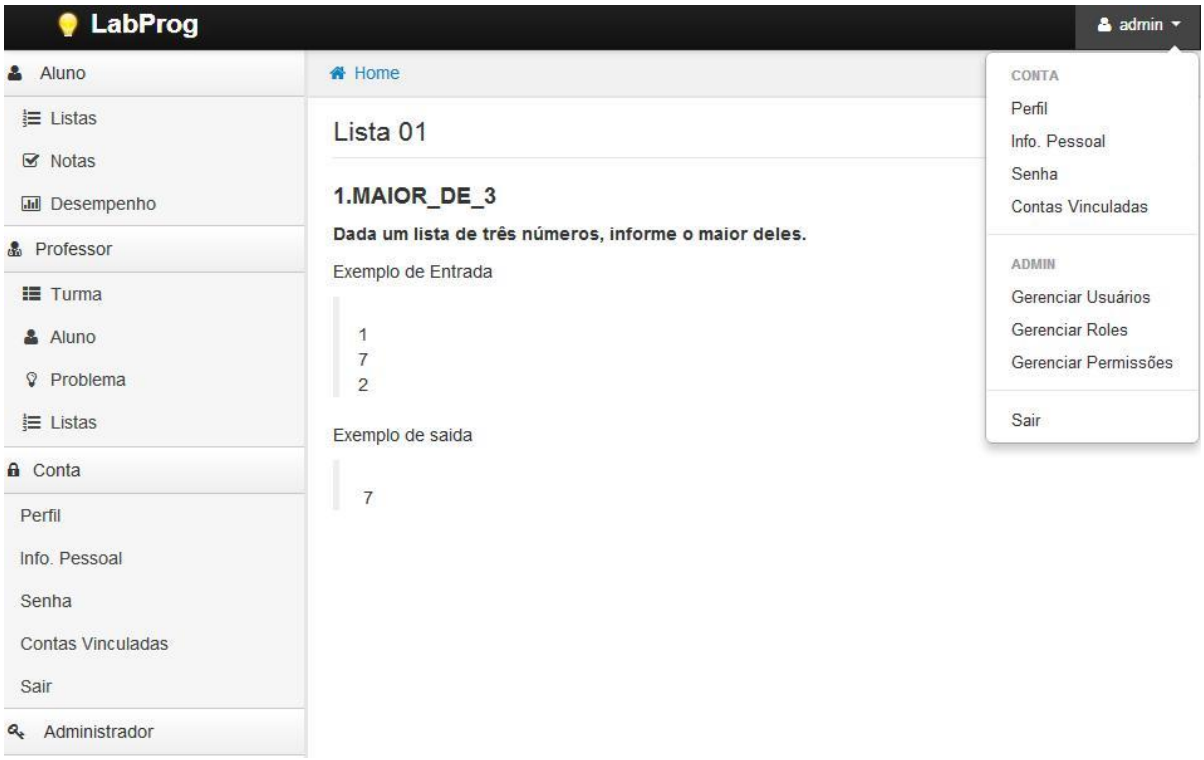

Figura 2. Interface do Ambiente LabProg.

Aliado a ferramenta web LabProg, está integrado um ambiente de desenvolvimento denominada NetBeans IDE (2015) usando a linguagem de programação JAVA. A integração resume-se a adição de um botão, que quando acionado realiza a identificação do estudante e qual problema pretende resolver. Neste ambiente de programação a resolução do problema é construída.

Dando sequência na interação, o IDE obtém as informações do problema, como os testes e as sugestões para auxiliar a resolução. Caso a resposta do aluno passe em todos os testes, o aluno é informado sobre o acerto, caso contrário ele apresenta uma sugestão, considerando a método de composição de resultado.

Todas as interações do estudante com o IDE de programação são registradas e enviadas para a ambiente web LabProg, para compor estatísticas e registrar os acertos e erros. Os testes isolam as entradas e resultados dos subproblemas, onde cada teste possui foco especifico, sendo possível identificar qual passo do método está.

Na elaboração de um problema o professor deve inserir a definição, exemplos de entrada e saída e informar os testes, que são entradas e saídas associadas para automatizar a correção, neste ponto ele pode associar sugestões aos testes, para que em uma eventual falha esse auxílio possa ser apresentado.

A elaboração dos casos de teste pode demandar bastante tempo e esforço por parte do professor, entretanto este se justifica pela utilização futura dos testes e problemas. Os testes não são exibidos, eliminando a tentativa de fraude dos estudantes tentam produzir o resultado sem responder o problema.

5. Experimento

Como forma de avaliar a viabilidade do uso prático do método de composição de resultado aliado ao ambiente de auxílio LabProg. Foi concebido um experimento para avaliar a aderência dos produtos deste estudo ao objetivo proposto.

Levando em consideração o que foi exposto, o método e as ferramentas possuem o foco de prover melhorias em atividades práticas. Portanto, os indivíduos que melhor se enquadram ao proposto, são estudante que já possuam conhecimento prévio de programação, e necessitam melhorar seu desempenho na resolução de problemas.

Para tanto, 39 alunos do curso do Ciência da Computação foram divididos em dois grupos aleatoriamente. O grupo A formado por 19 participantes sendo o grupo de controle, e o grupo B com 20 integrantes, sendo o grupo experimental, utilizaram o método e as ferramentas. Ambos os grupos possuíam condições semelhantes em relação a professores, laboratório e tempo de execução nas atividades. O delineamento experimental foi baseado em Queirós e Leal (2012).

As sessões de práticas de laboratório se deram em 7 encontros de uma hora e meia cada. Onde, em cada um foi apresentado uma Lista de 4 problemas. Totalizando 28 problemas e 10,5 horas de atividades. O primeiro encontro, em ambos os grupos, foi realizado um teste com questões de conhecimentos teóricos de programação e a execução prática da lista0, visando um diagnóstico inicial.

Para avaliar o desempenho entre os grupos foi utilizado a média de acertos de cada grupo ao longo dos encontros. Deste modo, foi considerada a comparação das medias de acerto dos grupos A e B, assim como a evolução do grupo experimental no decorrer das listas. No último encontro foi realizado um novo teste de conhecimentos teóricos e a lista6.

6. Analises

Dado o delineamento do item anterior, a figura 3 apresenta a evolução predominantemente superior do grupo experimental B em relação ao grupo A. onde são consideradas as médias de acertos dos grupos x pelas listas resolvidas nos encontros. Em quase todos os pontos de coleta o desvio padrão foi mensurado e considerado muito próximo entre os grupos.

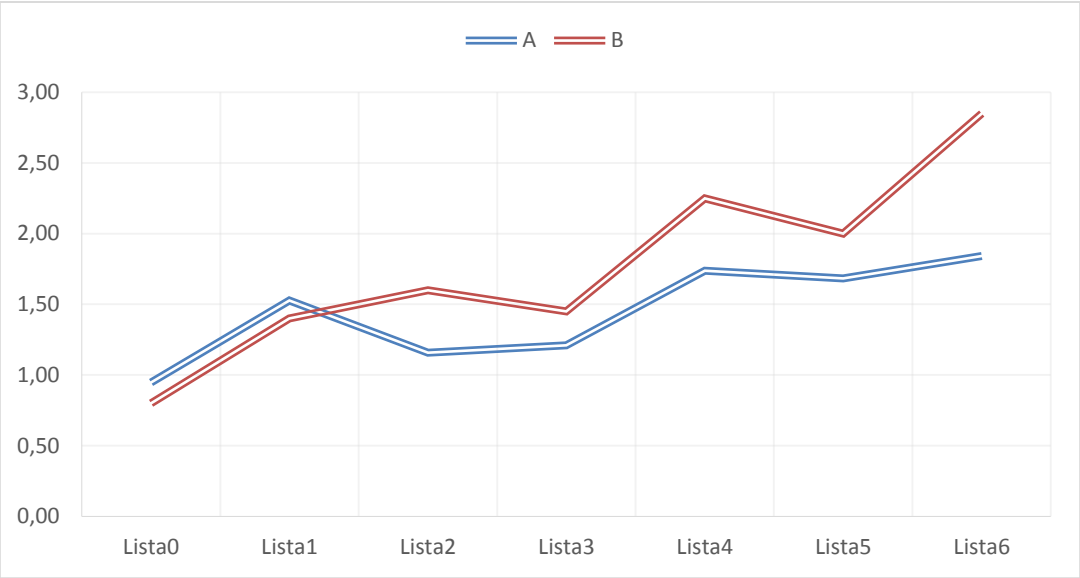

Figura 3. Evolução das médias de acertos A (controlado) e B (Experimental).

Os testes teóricos realizados durante o experimento não apresentaram diferença considerável entre os grupos no início e final. Já os testes práticos, representados na figura 4 apresenta o salto de desempenho entre os grupos considerando os encontros inicial e final, onde esse foi medido pelo média de aproveitamento na resolução dos problemas, sendo considerado o desempenho máximo de 100% o acerto de todas as atividades de uma lista de problemas.

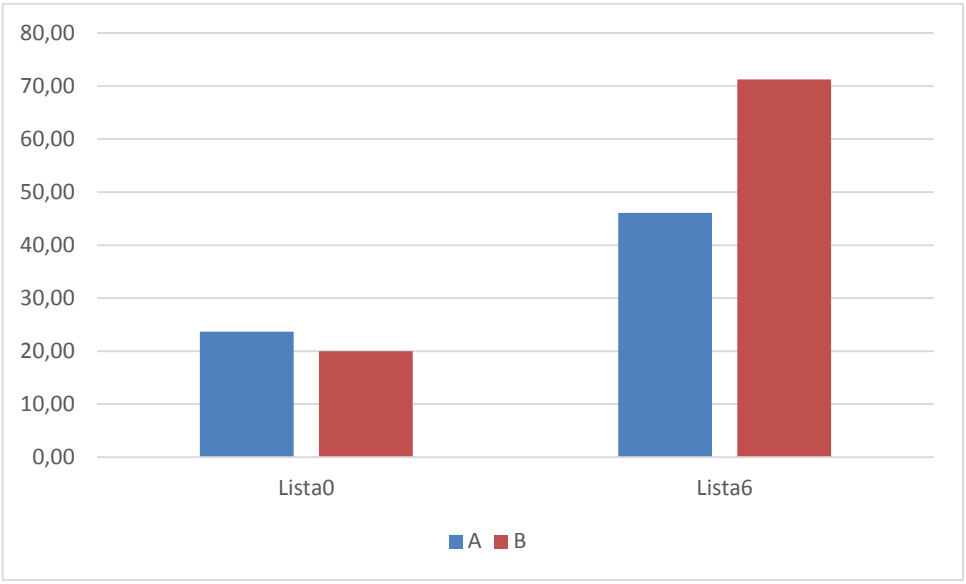

Figura 4. Comparação inicial e final das médias de acertos (100% => 4 problemas).

7. Considerações Finais

A aplicação do método e a utilização da ferramenta promovem melhorias no desempenho dos estudantes, considerando o grupo experimental. Tanto na evolução dos acertos, que considerando arredondamentos, inicialmente possuía a média de uma questão correta em ambos os grupos, passou a ser de dois e três nos grupos de controle e experimental respectivamente. Desta forma, expressando uma diferença de cerca de 30% na comparação inicial e final do aproveitamento dos estudantes nas listas de atividades.

A Metodologia de Resolução desenvolvida mostra-se promissora, visto que ela foi definida por meio de observações de como os estudantes resolviam os problemas, além de ser incrementada com experiências de desenvolvimento profissional. E como meio de potencializa-lo, foi agregada a correção automatizada que faz os testes de modo incremental, e ainda oferece sugestão de correção baseada no teste em que falhou.

O ferramental técnico desenvolvido apresenta inúmeras vantagens, para o professor provê o monitoramento em tempo real das atividades que estão sendo realizadas, desempenho dos estudantes e indicadores para intervenções. Como a revisão de temas relacionados aos erros mais frequentes. Para o estudante, proporciona uma experiência rica, em um ambiente de programação tanto educacional quanto profissional, utilizando a Método de Resolução para auxiliar a resposta de problemas e tendo feedbacks no decorrer das atividades.

Referências Bibliográficas

BECK, K. **Test-driven development: by example. 5. ed.** Boston: Addison-Wesley Professional, 2003.

BENNEDSEN, J.; CASPERSEN, M. E. Programming in context: a model-first approach to **CS1SIGCSE 2004.** Anais. 2004.

CORMEN, THOMAS H.; LEISERSON, CHARLES E.; RIVEST, RONALD L. C. S. **Algoritmos: Teoria e Prática. 3. ed**. Rio de Janeiro: CAMPUS, 2009.

COSTA, E. D. B. Raciocínio Baseado em Casos para auxılio a Alunos na Resolução de Problemas por Analogia – Uma abordagem para Representação e Recuperação de Casos. n. **Simpósio Brasileiro de Informática na Educação**, p. 593-602, 2008.

MOTA, M. P., BRITO, S. R., MOREIRA, M. P., FAVERO, E. L. Ambiente Integrado à Plataforma Moodle para Apoio ao Desenvolvimento das Habilidades Iniciais de Programação. In**: XX Simpósio Brasileiro de Informática na Educação**. SBC. 2009.

MOTA, MARCELLE PEREIRA; PEREIRA, LIS W. KANASHIRO; FAVERO, ELOI LUIZ. "JavaTool: Uma Ferramenta para Ensino de Programação". **Workshop de Educação em Computação**, Congresso anual da SBC. 2008.

NOBRE, I.; MENEZES, C. Suporte à Cooperação em um Ambiente de aprendizagem para Programação (SAmbA). **Simpósio Brasileiro de Informática na Educaçao**, n. SBIE, p. 337-347, 2002.

PELZ, F. D.; JESUS, E. A. DE; RAABE, A. L. A. Um Mecanismo para Correção Automática de Exercícios Práticos de Programação Introdutória. **Simpósio Brasileiro de Informática na Educação**, p. 26-30, 2012.

PROULX, V. K. Programming patterns and design patterns in the introductory computer science course**. Proceedings of the thirty-first SIGCSE technical symposium on Computer science education** - SIGCSE '00, p. 80-84, 2000.

QUEIRÓS, R.; LEAL, J. PETCHA: a programming exercises teaching assistant. **Proceedings of the 17th ACM annual conference**, p. 192–197, 2012.

ROCHA, P. S. ; B. Ferreira ; D. Monteiro . Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino. **Revista Novas Tecnologias na Educação**, RENOTE v. 9, p. 1, 2010.